**Amendments to the Claims**

Please amend claims 1-33; 35-37; 40-54; 56-64 all as shown below. All pending claims are reproduced below, including those that remain unchanged.

1. (Currently Amended) A computer-implemented system to marshal and unmarshal data between XML and ~~Java~~ an object-oriented programming language, comprising:

an XML data;

an XML schema which defines the XML data;

~~an XML~~ an object-oriented programming language type which corresponds to the XML schema and ~~implements a common Java type that~~ provides XML-oriented data manipulation, wherein the ~~XML~~ object-oriented programming language type allows the combination of XML and ~~Java~~ object-oriented programming language type systems and is capable of accessing and manipulating the XML data from within ~~Java~~ the object-oriented programming language; and

a compiler capable of generating the ~~XML~~ object-oriented programming language type from the XML schema, wherein the compiler runs on one or more processors.

2. (Currently Amended) The computer-implemented system according to claim 1, wherein:

the compiler is capable of generating the ~~XML~~ object-oriented programming language type based on the definition of a ~~Java~~ web service[[s]] method.

3. (Currently Amended) The computer-implemented system according to claim 1, wherein:

the compiler is capable of generating the ~~XML~~ object-oriented programming language type based on a definition file.

4. (Currently Amended) The computer-implemented system according to claim 1, wherein:

the compiler is capable of compiling [[a]] an ~~Java~~ object-oriented programming language project into one or more regular ~~Java~~ object-oriented programming language types.

5. (Currently Amended) The computer-implemented system according to claim 1, wherein: the ~~XML~~ object-oriented programming language type can be a movable cursor, capable of reading anywhere within the XML data.

6. (Currently Amended) The computer-implemented system according to claim 1, wherein: the ~~XML~~ object-oriented programming language type can be [[a]] an immovable value, capable of referencing a fixed part of the XML data.

7. (Currently Amended) The computer-implemented system according to claim 1, wherein: the ~~XML~~ object-oriented programming language type can be shared among multiple ~~Java~~ object-oriented programming language components.

8. (Currently Amended) The computer-implemented system according to claim 1, wherein: the ~~XML~~ object-oriented programming language type is capable of updating the XML data within ~~Java~~ the object-oriented programming language.

9. (Currently Amended) The computer-implemented system according to claim 1, wherein: the ~~XML~~ object-oriented programming language type is capable of accessing and updating ~~Java~~ object-oriented programming language data using ~~Java~~ object-oriented programming language type methods.

10. (Currently Amended) The computer-implemented system according to claim 1, wherein: the ~~XML~~ object-oriented programming language type is capable of accessing and updating a database.

11. (Currently Amended) The computer-implemented system according to claim 1, wherein:

the ~~XML~~ object-oriented programming language type is capable of a number of XML data operations, which include: querying XML data, transforming between XML types, and iterating over XML data document.

12. (Currently Amended) The computer-implemented system according to claim 1, further comprising:

an XML schema capable of defining the legal types of the XML data, which include constraints on data types and ranges of the XML data; and constraints on the data types and ranges of the ~~XML~~ object-oriented programming language type.

13. (Currently Amended) The computer-implemented system according to claim 12, wherein:

the compiler is capable of generating constraints on the ~~XML~~ object-oriented programming language type from the XML schema on legal types of the XML data.

14. (Currently Amended) The computer-implemented system according to claim 12, wherein:

the constraints on the ~~XML~~ object-oriented programming language type are capable of validating the ~~XML~~ object-oriented programming language type.

15. (Currently Amended) A computer-implemented system to transform types between XML and ~~Java~~ an object-oriented programming language, comprising:

~~a Java type~~ an XML data;

~~an XML~~ an object-oriented programming language type which ~~implements a common Java type that~~ provides XML-oriented data manipulation, wherein the ~~XML~~ object-oriented programming language type allows the combination of XML and ~~Java~~ object-oriented programming language type systems and is capable of accessing the XML data from within ~~Java~~

the object-oriented programming language without mapping the XML data to an ~~Java~~ object-oriented programming language object; and

an XML transformation capable of transforming a source type to a target type, wherein the source and target ~~type~~ types can be either [[the]] an XML type or [[the]] an ~~Java~~ object-oriented programming language type.

16. (Currently Amended) The computer-implemented system according to claim 15, further comprising:

a global registry of XML transformations capable of looking up an existing XML transformation between a source and a target type.

17. (Currently Amended) The computer-implemented system according to claim 15, further comprising:

a library of XML transformations capable of looking up an existing XML transformation by name between a source and a target type.

18. (Currently Amended) A computer-implemented system to marshal and unmarshal data between XML and ~~Java~~ an object-oriented programming language, comprising:

an XML data;

an XML schema which defines the XML data;

a lightweight XML store capable of retaining the XML data as a searchable index; and

~~an XML~~ an object-oriented programming language type which corresponds to the XML schema and ~~implements a common Java type that~~ provides XML-oriented data manipulation, wherein the ~~XML~~ object-oriented programming language type allows the combination of XML and ~~Java~~ object-oriented programming language type systems and is capable of referencing the lightweight XML store and accessing elements of the XML data from within ~~Java~~ the object-oriented programming language.

19. (Currently Amended) A computer-implemented system to marshal and unmarshal data between XML and ~~Java~~ an object-oriented programming language, comprising:

an XML data;

an XML schema which defines the XML data;

a lightweight XML store capable of retaining the XML data at the text or tag level; and ~~an XML~~ an object-oriented programming language type which corresponds to the XML schema and ~~implements a common Java type that~~ provides XML-oriented data manipulation, wherein the ~~XML~~ object-oriented programming language type allows the combination of XML and object-oriented programming language type systems and is capable of referencing the lightweight XML store and accessing elements of the XML data from within ~~Java~~ the object-oriented programming language.

20. (Currently Amended) The computer-implemented system according to claim 19, wherein:

the lightweight XML store is capable of representing the retained XML data as a hierarchical structure.

21. (Currently Amended) The computer-implemented system according to claim 20, wherein:

the hierarchical structure can be a tree.

22. (Currently Amended) The computer-implemented system according to claim 19, wherein:

the ~~XML~~ object-oriented programming language type is capable of accessing the XML data incrementally.

23. (Currently Amended) A method to marshal and unmarshal data between XML and ~~Java~~ an object-oriented programming language, comprising:

defining an XML data using an XML schema;

accessing from within ~~Java~~ object-oriented programming language elements of the XML data via ~~an XML~~ an object-oriented programming language type which corresponds to the XML schema and ~~implements a common Java type that~~ provides XML-oriented data manipulation, wherein the ~~XML~~ object-oriented programming language type allows the combination of XML and ~~Java~~ object-oriented programming language type systems; and

generating the ~~XML~~ object-oriented programming language type from the XML schema using a compiler.

24. (Currently Amended) The method according to claim 23, further comprising:

generating the ~~XML~~ object-oriented programming language type based on the definition of a ~~Java~~ web ~~services~~ service method.

25. (Currently Amended) The method according to claim 23, further comprising:

generating the ~~XML~~ object-oriented programming language type based on a definition file.

26. (Currently Amended) The method according to claim 23, further comprising:

compiling a Java project into one or more regular ~~Java~~ object-oriented programming language types.

27. (Currently Amended) The method according to claim 23, further comprising:

utilizing the ~~XML~~ object-oriented programming language type as a movable cursor to read anywhere within the XML data.

28. (Currently Amended) The method according to claim 23, further comprising:[[:]] utilizing the ~~XML~~ object-oriented programming language type as [[a]] an immovable value to reference a fixed part of the XML data

29. (Currently Amended) The method according to claim 23, further comprising: sharing the ~~XML~~ object-oriented programming language type among multiple ~~Java~~ object-oriented programming language components.

30. (Currently Amended) The method according to claim 23, further comprising: updating the XML data within ~~Java~~ an object-oriented programming language via the ~~XML~~ object-oriented programming language type.

31. (Currently Amended) The method according to claim 23, further comprising: accessing and updating ~~Java~~ object-oriented programming language data using ~~Java~~ object-oriented programming language type methods.

32. (Currently Amended) The method according to claim 23, further comprising: accessing and updating a database via the ~~XML~~ object-oriented programming language type.

33. (Currently Amended) The method according to claim 23, further comprising: utilizing a number of XML data operations via ~~XML~~ the object-oriented programming language type, these operations include: querying XML data, transforming between XML types, and iterating over XML data document.

34. (Original) The method according to claim 23, further comprising:

defining the legal types of the XML data via an XML schema, which include constraints on data types and ranges of the XML data.

35. (Currently Amended) The method according to claim 34, further comprising:

generating constraints on the data types and ranges of the ~~XML~~ object-oriented programming language type from the XML schema on legal types of the XML data.

36. (Currently Amended) The method according to claim 34, further comprising:

validating the ~~XML~~ object-oriented programming language type using the constraints on the ~~XML~~ object-oriented programming language type.

37. (Currently Amended) A method to transform types between XML and ~~Java~~ an object-oriented programming language, comprising:

utilizing a ~~Java type~~ XML schema;

utilizing ~~an XML~~ an object-oriented programming language type which corresponds to the XML schema and ~~implements a common Java type that~~ provides XML-oriented data manipulation, wherein the ~~XML~~ object-oriented programming language type allows the combination of XML and ~~Java~~ object-oriented programming language type systems and is capable of accessing and manipulating an XML data that is defined by the XML schema from within ~~Java~~ the object-oriented programming language; and

transforming a source type to a target type via an XML transformation, wherein the source and target types can be either [[the]] an XML type or [[the]] an ~~Java~~ object-oriented programming language type.

38. (Original) The method according to claim 37, further comprising:

looking up an existing XML transformation between a source and a target type via a global registry of XML transformations.

39. (Original) The method according to claim 37, further comprising:

looking up an existing XML transformation by name between a source and a target type via a library of XML transformations.

40. (Currently Amended) A computer-implemented method to marshal and unmarshal data between XML and ~~Java~~ an object-oriented programming language, comprising:

retaining an XML data as a searchable index via a lightweight XML store; and

referencing the lightweight XML store and accessing from within ~~Java~~ the object-oriented programming language the XML data via [[the]] ~~XML~~ an object-oriented programming language type which corresponds to the XML schema and ~~implements a common Java type that~~ provides XML-oriented data manipulation, wherein the ~~XML~~ object-oriented programming language type allows the combination of XML and ~~Java~~ object-oriented programming language type systems is capable of accessing and manipulating the XML data.

41. (Currently Amended) A computer-implemented method to marshal and unmarshal data between XML and ~~Java~~ an object-oriented programming language comprising:

retaining an XML data at the text or tag level via a lightweight XML store; and

referencing the lightweight XML store and accessing from within ~~Java~~ the object-oriented programming language the XML data via [[the]] ~~XML~~ an ~~Java~~ object-oriented programming language type which is corresponding to an XML schema that defines the XML data and ~~implements a common Java type that~~ provides XML-oriented data manipulation, wherein the ~~XML~~ object-oriented programming language type allows the combination of XML and ~~Java~~ object-oriented programming language type systems and is capable of accessing elements of the XML data.

42. (Currently Amended) The computer-implemented method according to claim 41, further comprising:

representing the retained XML data as a hierarchical structure, which can be a tree.

43. (Currently Amended) The computer-implemented method according to claim 41, further comprising:

accessing the XML data incrementally via the ~~XML~~ object-oriented programming language type.

44. (Currently Amended) A machine readable storage medium having instructions stored thereon that when executed by a processor cause a system to:

define an XML data using an XML schema;

access the XML data via ~~an XML~~ an object-oriented programming language type which corresponds to the XML schema and ~~implements a common Java type that~~ provides XML-oriented data manipulation, wherein the ~~XML~~ object-oriented programming language type allows the combination of XML and ~~Java~~ object-oriented programming language type systems and is capable of accessing and manipulating the XML data from within ~~Java~~ an object-oriented programming language; and

generate the ~~XML Java~~ object-oriented programming language type from the XML schema using a compiler.

45. (Currently Amended) The machine readable storage medium of claim 44, further comprising instructions that when executed cause the system to:

generate the ~~XML~~ object-oriented programming language type based on the definition of a ~~Java~~ web service[[s]] method.

46. (Currently Amended) The machine readable storage medium of claim 44, further comprising instructions that when executed cause the system to:

generate the ~~XML~~ object-oriented programming language type based on a definition file.

47. (Currently Amended) The machine readable storage medium of claim 44, further comprising instructions that when executed cause the system to:

compile ~~a Java~~ an object-oriented programming language project into one or more regular ~~Java~~ object-oriented programming language types with the compiler.

48. (Currently Amended) The machine readable storage medium of claim 44, further comprising instructions that when executed cause the system to:

utilize the ~~XML~~ object-oriented programming language type as a movable cursor to read anywhere within the XML data.

49. (Currently Amended) The machine readable storage medium of claim 44, further comprising instructions that when executed cause the system to:

utilize the ~~XML~~ object-oriented programming language type as an immovable value to reference a fixed part of the XML data.

50. (Currently Amended) The machine readable storage medium of claim 44, further comprising instructions that when executed cause the system to:

share the ~~XML~~ object-oriented programming language type among multiple ~~Java~~ object-oriented programming language components.

51. (Currently Amended) The machine readable storage medium of claim 44, further comprising instructions that when executed cause the system to:

update the XML data within ~~Java~~ an object-oriented programming language via the ~~XML~~ object-oriented programming language type.

52. (Currently Amended) The machine readable storage medium of claim 44, further comprising instructions that when executed cause the system to:

access and update ~~Java~~ object-oriented programming language data using regular ~~Java~~ object-oriented programming language type methods.

53. (Currently Amended) The machine readable storage medium of claim 44, further comprising instructions that when executed cause the system to:

access and update a database via the ~~XML~~ object-oriented programming language type.

54. (Currently Amended) The machine readable storage medium of claim 44, further comprising instructions that when executed cause the system to:

utilize a number of XML data operations via the ~~XML~~ object-oriented programming language type, these operations include:

querying XML data, transforming between XML types, and

iterating over XML data document.

55. (Original) The machine readable storage medium of claim 44, further comprising instructions that when executed cause the system to:

define the legal types of the XML data via an XML schema, which include constraints on data types and ranges of the XML data.

56. (Currently Amended) The machine readable storage medium of claim 55, further comprising instructions that when executed cause the system to:

generate constraints on the ~~XML~~ object-oriented programming language type from the XML schema on legal types of the XML data.

57. (Currently Amended) The machine readable storage medium of claim 55, further comprising instructions that when executed cause the system to:

validate the ~~XML~~ object-oriented programming language type using the constraints on the ~~XML~~ object-oriented programming language type.

58. (Currently Amended) A machine readable storage medium having instructions stored thereon that when executed by a processor cause a system to:

utilize a ~~Java type~~ XML schema;

utilize ~~an XML~~ an object-oriented programming language type which corresponds to the XML schema and ~~implements a common Java type that~~ provides XML-oriented data manipulation, wherein the ~~XML~~ object-oriented programming language type allows the combination of XML and ~~Java~~ object-oriented programming language type systems and is capable of accessing and manipulating an XML data that is defined by the XML schema from within ~~Java~~ an object-oriented programming language; and

transform a source type to a target type via an XML transformation, wherein the source and target type can be either [[the]] an XML type or [[the]] an object-oriented programming language type.

59. (Currently Amended) The machine readable storage medium of claim 58, further comprising instructions that when executed cause the system to:

look up an existing XML transformation between a source and a target type via a global registry of XML transformations.

60. (Currently Amended) The machine readable storage medium of claim 58, further comprising instructions that when executed cause the system to:

look up an existing XML transformation by name between a source and a target type via a library of XML transformations.

61. (Currently Amended) A machine readable storage medium having instructions stored thereon that when executed by a processor cause a system to:

retain an XML data as a searchable index via a lightweight XML store; and

reference the lightweight XML store and access from within ~~Java~~ an object-oriented programming language the XML data via [[the]] ~~XML~~ an object-oriented programming language type which corresponds to [[the]] an XML schema and ~~implements a common Java type that~~ provides XML-oriented data manipulation, wherein the ~~XML Java~~ object-oriented programming language type allows the combination of XML and ~~Java~~ object-oriented programming language type systems and is capable of accessing and manipulating the XML data.

62. (Currently Amended) A machine readable storage medium having instructions stored thereon that when executed by a processor cause a system to:

retain an XML data at the text or tag level via a lightweight XML store; and

reference the lightweight XML store and access from within ~~Java~~ an object-oriented programming language the XML data via [[the]] ~~XML~~ an object-oriented programming language type which corresponds to [[the]] an XML schema and ~~implements a common Java type that~~ provides XML-oriented data manipulation, wherein the ~~XML~~ object-oriented programming language type allows the combination of XML and ~~Java~~ object-oriented programming language type systems and is capable of accessing and manipulating the XML data.

63. (Currently Amended) The machine readable storage medium of claim 62, further comprising instructions that when executed cause the system to:

represent the retained XML data as a hierarchical structure, which can be a tree.

64. (Currently Amended) The machine readable storage medium of claim 62, further comprising instructions that when executed cause the system to:
access the XML data incrementally via the XML type.

65 - 66. (Canceled)